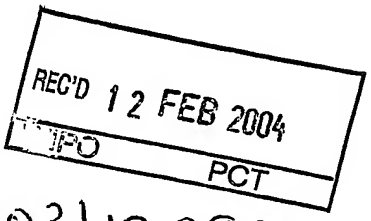
EP03/12098

BUNDESREPUBLIK DEUTSCHLAND

#2

19. 11. 2003

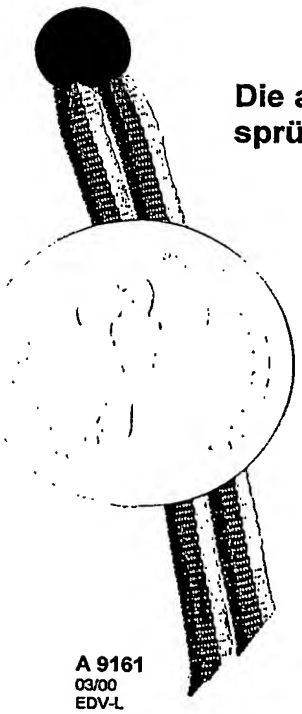
PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)



EP 03 12098

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen:	102 53 275.3
Anmeldetag:	15. November 2002
Anmelder/Inhaber:	Siemens Aktiengesellschaft, München/DE
Bezeichnung:	Verfahren zur Erzeugung eines Bitstroms aus einem Indizierungsbaum
IPC:	H 04 N 7/24



Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 10. November 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Schmidt C.

A 9161
03/00
EDV-L

Beschreibung

Verfahren zur Erzeugung eines Bitstroms aus einem Indizierungsbaum

1. Problem

Diese Erfindungsmeldung betrifft die Problemstellung zu Bestimmen ob Inhalte entsprechend einer vorab getroffenen Abfrage in einem Bitstrom enthalten sind und/oder solche Inhalte gezielt aus Bitstrom auszulesen, ohne den gesamten Bitstrom parsen zu müssen. Eine solche Abfrage kann beispielsweise in Form einer Abfrage formuliert in einer Abfragesprache wie beispielsweise SQL[1] oder XPath[2] vorliegen oder in Form einer Filterbedingung vorliegen.

Um die gesuchten Informationen zu finden und/oder zu bestimmen ob diese enthalten sind kann der gesamte Bitstrom durchsucht werden. Eine weitere Möglichkeit ergibt sich, wenn bei dem Senden des Bitstroms vorab oder an bestimmten Stellen im Bitstrom Indexdaten abgelegt werden, die den Inhalt im Bitstrom indexieren. Im letzten Fall genügt es nur die Indexdaten zu durchsuchen. Die bisherigen Lösungen in diesen zwei Kategorien werden im Abschnitt 2 erläutert. Die bekannten Lösungen erfordern jedoch die gesamten Indexdaten von Anfang bis zu den Daten zu durchsuchen, die der Abfrage entsprechen. Zudem sind die Indexdaten, die im Bitstrom zu übertragen sind, sehr groß.

Abschnitt 3 beschreibt die erfindungsmäßige Lösung des Problems.

Als ein Beispiel dieser Problemstellung wird im Folgenden davon ausgegangen, dass ein Beschreibungselement aus einem MPEG-7 Beschreibungsstrom auszulesen ist. Eine MPEG7-Beschreibung [3] wird dabei mit Hilfe der Extensible Markup Language (XML) erstellt. Eine Beschreibung kann in mehrere Einheiten (Access Units) aufgeteilt sein, welche ihrerseits wiederum aus mehreren Fragmenten (Fragment Update Units) be-

stehen [3]. Diese Einheiten werden dann encodiert und bei Bedarf als MPEG-7 Beschreibungsstrom an einen oder mehrere Empfänger versandt.

Es gibt bereits eine Vielzahl von Abfragesprachen, die es erlauben, nach bestimmten Informationen in einem XML-Dokument zu suchen. Im folgenden wird beispielsweise angenommen, dass eine Abfrage in der Abfragesprache XPath [2] vorliegt, mit der ein Element aus einem MPEG-7 Beschreibungsstrom abgerufen werden soll.

Es können nun Szenarien auftreten, wo ein Empfänger mehrere MPEG7-Beschreibungen erhält. Nicht alle davon sind für den jeweiligen von Nutzen. Mit Hilfe von XPath können Auswahlkriterien zur Filterung gewünschter Informationen innerhalb einer Beschreibung definiert werden. Ziel einer Abfrage kann einerseits die Bewertung sein, ob eine entsprechende Einheit für den Empfänger wichtig ist oder nicht. Anderseits kann mit Hilfe einer Abfrage gezielt auf gewünschte Informationen innerhalb eines MPEG-7 Beschreibungsstroms zugreifen.

Das beschriebene Problem tritt aber nicht nur im Kontext von MPEG7 auf. Die hier präsentierte Lösung lässt sich im Allgemeinen auf Datenströme anwenden, in denen Informationen indiziert werden sollen.

2. Bisherige Lösungen:

MPEG7 [3] hat bis jetzt für den wahlfreien Zugriff auf bestimmte Elemente bzw. für die Filterung von Metadaten ohne parsen des gesamten Datenstroms keine Mechanismen vorgesehen. Das bedeutet, dass auf der Empfängerseite jede empfangene Einheit zunächst betrachtet und in speziellen Fällen decodiert werden muss. Dadurch erhält man wieder eine Beschreibung in XML-Format. Über den XML-Text wird dann eine XPath-Abfrage ausgeführt, um gewünschten Informationen zu enthalten bzw. zu entscheiden, ob gewisse Kriterien enthalten sind oder nicht. Das Decodieren und dann Abarbeiten des XML-Textes ist sehr zeitaufwendig und daher in zeitkritischen Fällen inak-

zeptabel. Zudem kann es sein, dass der Speicher im Endgerät begrenzt ist, was dazu führen kann, dass Beschreibungen nicht vollkommen decodiert werden können. Verläuft eine XPath-Abfrage negativ, wurde der Aufwand des Decodieren umsonst betrieben.

Im Rahmen von TV-Antytime (TVA) [4] wurde eine Index-Struktur vorgestellt, die einen wahlfreien Zugriff auf bestimmte Elemente innerhalb eines Fragmentes erlaubt. Diese Index-Struktur besteht aus mehreren Teilen. Einstiegspunkt ist eine Liste (Key Index List), in der sämtliche indizierte Pfade eines Dokumentes abgelegt werden. Bei einer Abfrage werden diese Pfade der Reihe nach mit der Abfrage verglichen, bis ein entsprechender Eintrag gefunden wird. Durch die Informationen, die zu diesem Eintrag gespeichert werden, können die Stellen im Strom bestimmt werden an denen das indexierte Element codiert vorliegt.

Hierzu wird eine weitere Indexstruktur verwendet, die auch die Suche nach Elementwerten erlaubt. Dies ist jedoch für diese Erfindungsmeldung nicht erheblich bzw. kann entsprechend auch mit dem erfindungsmäßigen Verfahren realisiert werden.

Die beschriebene Lösung vermeidet durch die „Key Index List" das Decodieren von uninteressanten Fragmenten, was sich auch positiv auf den benötigten Speicherplatz während der Suche im Empfänger auswirkt. Das lineare Durchwandern der Key-Index-List ist aber nach wie vor ein zeitkonsumierender Faktor. Zudem ist die Übertragung aller indexierter Pfade aufwendig.

3. Problemlösung

In diesem Beispiel soll auf das erfindungsmäßige Verfahren eingegangen werden, einen Indexierungsbaum in einem Bitstrom effizient zu übertragen, so dass er ohne Decodierung vom Empfänger zur Indexierung genutzt werden kann.

3.A) Indexierungsstruktur

In diesem Erfindungsmäßigen Verfahren werden

- die Indexdaten bezüglich eines Schlüssels in einer Baumstruktur sortiert,
- die Indexdaten in der Baumstruktur zunächst der Tiefe nach („Depth First") in den Bitstrom eingefügt,
- zu den Indexdaten Sprunginformation abgespeichert, um nicht relevante Informationen im Bitstrom überspringen zu können ohne sie linear durchsuchen zu müssen und
- der so generierte Bitstrom zum Empfänger übertragen.

An welchen Stellen die Sprunginformationen eingefügt werden, soll im Folgenden in einer Ausprägung dieses erfindungsmäßigen Verfahrens anhand eines B-Baums erläutert werden.

3.A.1) Ausführung

Der B-Baum ist eine weit verbreitete Datenstruktur eines Indexbaumes, welcher häufig bei der Indizierung von Daten zur Anwendung kommt. Dazu wird zunächst der Baum durch Einfügen verschiedener Sucheinträge (sogenannter Schlüssel) aufgebaut. Da der B-Baum eine ausgeglichene Struktur aufweist, wird ein Suchen mit einem geringeren, im Vergleich zur Liste nur logarithmischem Aufwand garantiert.

Hiermit kann das lineare prozessieren aller Indexeinträge, welches bei dem Konzept einer List wie z.B. bei TVA erforderlich ist, durch den Einsatz eines B-Baumes vermieden werden. Als Schlüssel S werden bei diesem Beispiel die XPATH-Pfade von dem Wurzelknoten zu den einzelnen indexierten Dokumentknoten eines XML Beschreibungsbaumes bzw. Bitstroms verwendet. Als indexierte Dokumentknoten werden in dieser Ausarbeitung alle Blattknoten d.h. alle XML Elemente mit einfachem Inhalt und alle XML Attribute verwendet. Demnach sind im Index bei diesem Beispiel alle Wurzel-Blatt-Pfade eines bestimmten XML Beschreibungsbaumes enthalten. In dem Indexbaum I werden nur unterschiedliche Schlüssel (XPATH-Pfade) S abgelegt. Daher wird zusätzlich zum Schlüssel S selber im Eintrag

E eines Indexbaums I mitgespeichert, wie oft ein Schlüssel S im Dokument auftritt und wo im Datenstrom jeweils dieser indexierte Dokumentknoten gespeichert ist. Jeder Knoten KI im Indexbaum kann mehrere Einträge EI haben, die gemäß ihrem Schlüssel S sortiert sind (s. Abb. 1). Vor und nach jedem Eintrag EI_a in einem Knoten im Indexbaum I kann eine Referenz auf einen Kindknoten gespeichert werden, die Einträge EI_b beinhalten, die gemäß ihrem Schlüssel und der Ordnung vor bzw. nach dem Eintrag EI_a anzuordnen sind. Bei einer Abfrage kann anhand der Einträge im Vaterknoten bestimmt werden welcher der Kinderknoten einen passenden Eintrag bzw. Schlüssel enthalten kann. Demnach müssen nicht alle Kinderknoten, sondern nur einer durchsucht werden.

Auf den exakten Aufbau des B-Baumes als Indexbaum wird hier nicht genauer eingegangen, da der B-Baum eine bereits weit verbreitete Datenstruktur ist und in der Literatur daher bereits hinlänglich beschrieben wurde[5]. In diesem Beispiel soll auf das erfindungsmäßige Verfahren eingegangen werden, einen solchen Indexierungsbaum in einem Bitstrom zu übertragen, so dass er ohne Decodierung vom Empfänger genutzt werden kann.

Um die geringere, im Vergleich zur Indexierungsliste nur logarithmische Zugriffskomplexität auch bei Übertragung der Indexstruktur im Datenstrom zu ermöglichen wird der B-Baum gemäß dem erfindungsmäßigen Verfahren in „Depth-First" [6] Ordnung in den Datenstrom eingefügt. Hierbei werden die Einträge und die relativen Sprungadressen der Kindknoten abgespeichert. Lediglich die Referenz auf den ersten Kindknoten wird nicht abgespeichert, da dieser im Anschluss an den Vaterknoten übertragen wird. Enthält ein Knoten im B-Baum mehrere Kindknoten, so werden außer für den ersten Kindknoten die relative Sprungadresse an den Speicherplatz aller anderen Kindknoten im Strom angegeben. Der erste Kindknoten wird nach diesen Sprungadressen in den Datenstrom eingefügt. Mit den Sprunginformationen, die beispielsweise als relative Speicheradressen des Bitstroms abgelegt sind, kann direkt auf die

entsprechenden Indexeinträge im Baum zugegriffen werden, die noch der Abfrage gleichen könnten. Diese Methode ermöglicht nicht nur eine eindeutige Rekonstruktion des B-Baumes aus dem Strom, sondern erlaubt es auch, dass nicht benötigte Knoteninformationen einfach übersprungen werden können. Dadurch kann das Einlesen der Indexinformationen beschleunigt werden und somit ist die Suche effizienter.

Anhand eines konkreten Beispiels sollen obige Ausführungen näher erläutert werden. Zunächst wurden alle vollständigen Pfade eines bestimmten Dokumentes generiert, welche dann in einen B-Baum eingetragen wurden. Teile des resultierenden Baumes sind in Abb. 1 dargestellt.

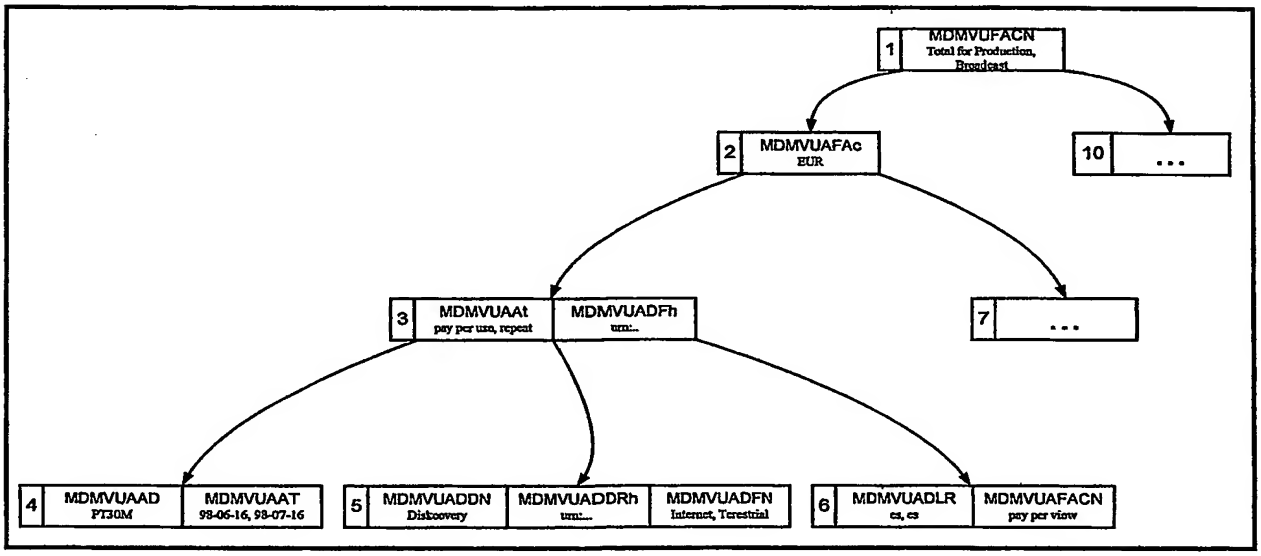


Abb. 1: Beispiel eines Indexbaums, bei dem die Knoten nach „Depth first" durchnummeriert sind.

Um den Teilbaum übersichtlicher zu halten, wurden dabei für Elementnamen nur die Anfangsbuchstaben verwendet. So steht MDMVUFACN beispielsweise für den Pfad Mpeg7/Description/MultimediaContent/Video/UsageInformation/FinancialResults/AccountItem/CostType/Name. In dem Beispielsdokument tritt dieser Pfad zweimal auf, einmal mit der Werteausprägung „Total for Production" und ein zweites Mal mit dem Wert „Broadcast".

Nachdem der B-Baum vollständig aufgebaut wurde, wird er dann wie oben beschrieben auf einen linearen Strom abgebildet. Die

Reihenfolge in der die Knoten abgebildet werden wird in der obigen Graphik durch die Knotennummern signalisiert.

Dabei werden pro Knoten folgende Informationen gespeichert:

Knoten {
Anzahl der Einträge
für jeden Eintrag:
Schlüssel
Anzahl der Instanziierungen
für jede Instanziierung:
Werteausprägung
Position
für jeden Kindknoten außer den ersten (Anzahl der Schlüssel)
Offset im Strom

Für den oben abgebildeten B-Baum würde der Beginn des Stroms wie folgt aussehen:

1	1, MDMVUAFCN, 2, Total for Production, **position**, Broadcast, , **position**, offset(10)	**2**	...

3	2, MDMVUAAt, 2, pay per use, **position**, repeat, **position**, MDMVUADFh, 1, urn:.., **position**, offset(5), offset(6)

4	...	...

Die Knotennummern sind in der Graphik nur übersichtshalber aufgeführt, werden aber nicht mit dem Strom übermittelt.

Nachdem der Empfänger nun einen solchen Strom erhalten hat, werden nach und nach die benötigten Knoten entsprechend der Abfrage aus dem Strom ausgelesen. Durch Vergleiche zwischen

der Abfrage und den Schlüsselwerten werden solange bestimmte Knoteninformationen aus dem Strom ausgelesen, bis der entsprechende Eintrag gefunden wurde oder bis keine weiteren passenden Einträge mehr vorhanden sind. In letzteren Fall ist die gewünschte Information im indizierten Dokument nicht enthalten.

Für eine Beispielsabfrage Mpeg7/Description/MultimediaContent/Video/UsageInformation/Availability/Dissemination/Disseminator/Agent/Name = „Discovery" (kurz MDMVUADDAN = „Discovery" werden zunächst die Informationen des ersten Knotens ausgelesen. Da der Suchstring alphabetisch gesehen kleiner ist, als der einzige Schlüsselwert im ersten Knoten wird auf den zweiten Knoten zugegriffen. Dementsprechend muss nicht der gesamte Baum durchsucht werden sondern nur entlang einem Pfad im Baum. In Abb. 2 ist der Pfad, der bei dieser Abfrage durch den Indexbaum in Abb. 1 durchlaufen wird, dargesetllt.

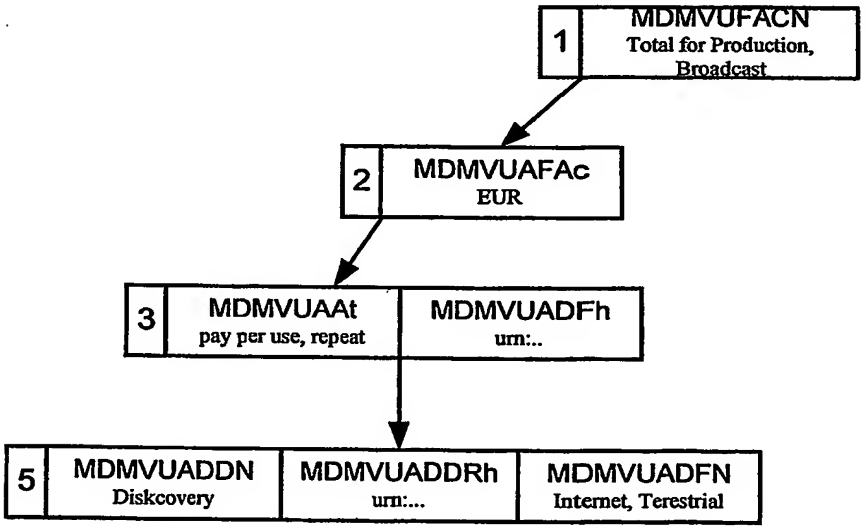


Abb. 2

Im fünften Knoten wird der passende Wert gefunden und man erhält dann die Position des entsprechenden Elementes im indizierten Dokument. Alle anderen Knoten des ursprünglichen B-Baumes werden im Strom ignoriert bzw. übersprungen ohne sie zu betrachten.

3.B) Schlüsselwerte:

In einer weiteren Ausprägung des in 3.A) beschriebenen, erfindungsmäßigen Verfahrens werden als Schlüssel XPATH Pfade verwendet.

3.C) Relative Schlüsselwerte:

In einer weiteren Ausprägung des in 3.B) beschriebenen, erfindungsmäßigen Verfahrens werden als Schlüssel relative XPATH Pfade in Bezug auf den im vorangegangen übertragenen Schlüssel im Vaterknoten und/oder dem im vorangegangen übertragenen Schlüssel im selben Knoten des Indexbaumes verwendet.

3.C.1) Ausführung

Bei dem in 3.B beschriebenen Verfahren werden die XPATH Pfade, die als Schlüssel im Indexbaum dienen, mit ihrer Bytedarstellung im Strom abgelegt. Wie aus dem Beispiel hervorgeht, können diese Einträge aus sehr vielen Zeichen bestehen. Um das Datenvolumen im Strom zu verringern, können die Gemeinsamkeiten in den Pfaden ausgenutzt werden. Pfade innerhalb eines Knotens bzw. Pfade in Bezug auf den Vaterknoten haben meist einen gemeinsamen Anteil (Prefix). Es genügt also, diesen Prefix nur einmal zu übertragen und die Pfade relativ zu ihren Nachbarschlüssel im Knoten bzw. Schlüssel im Vaterknoten zu übermitteln.

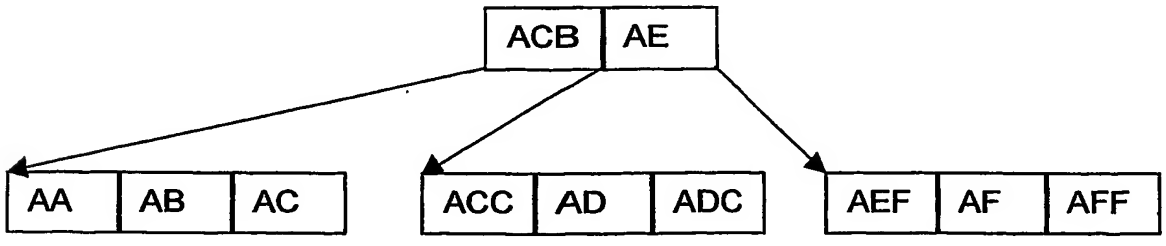


Abb. 3

In Abb. 3 ist hierzu ein vereinfachter Indexbaum mit Schlüsseln angegeben, der lexikographisch geordnet ist. Für die relative Adressierung wird nun zunächst in jedem Knoten ausgehend von dem ersten Schlüssel die restlichen relativ zum vorhergehenden codiert. In Abb. 4 ist dies veranschaulicht, wobei in der Darstellung „." einen Schritt zum Vater im XPATH-Pfad repräsentiert [2].

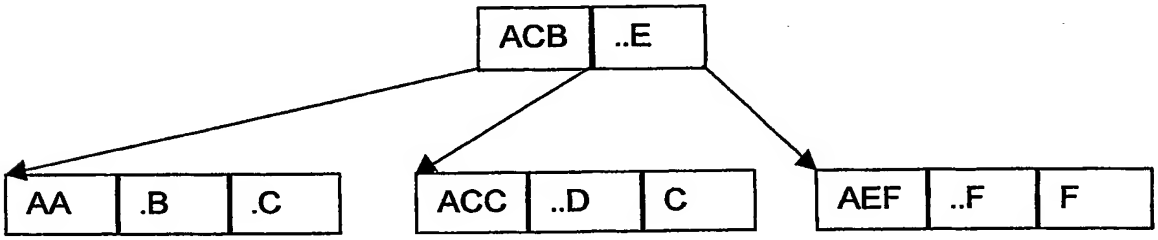


Abb. 4

In Abb. 5 werden nun die ersten Schlüssel der Kindknoten relativ zu dem Vorangegangenem Schlüssel im Vaterknoten des Indexbaumes codiert. Für den ersten Schlüssel (AA s. Abb. 3) des ersten Kindknotens wird ebenso wie für den (ACC s. Abb. 3) des zweiten Kindknotens wird der erste Schlüssel (ACB s. Abb. 3) des Vaterknotens verwendet.

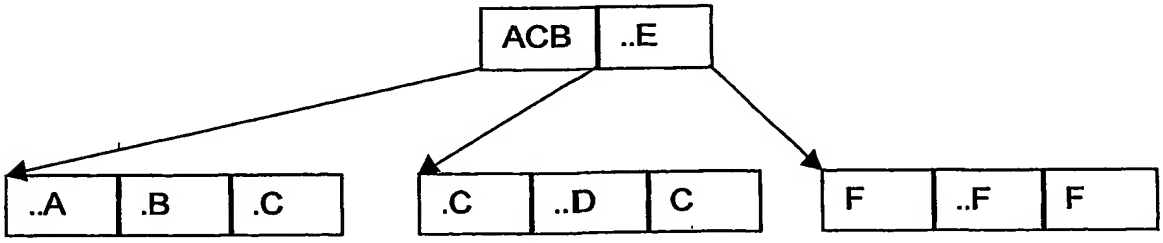


Abb. 5

3.D) Sortierte relative Schlüsselwerte:

In einer weiteren Ausprägung des in 3.C) beschriebenen, erfindungsmäßigen Verfahrens werden als Schlüssel relative XPATH Pfade verwendet, die sich auf den letzten übertragenen Schlüssel des Vaterknotens des Indexbaumes im Bitstrom beziehen, und bei dem die Einträge im ersten Kindknoten eines Vaterknotens im Indexbaum in umgekehrter Reihenfolge zu den Einträgen der anderen Kindknoten übertragen werden.

3.D.1) Ausführung

Die durch relative XPATH Pfade eingesparte Datenmenge kann maximiert werden, indem der XPATH Pfad relativ bezüglich dem nächsten Schlüsselwert im Indexbaum codiert wird. Der Indexbaum wird dementsprechend im Strom sortiert. In Abb. 3 ist hierzu ein vereinfachter Indexbaum mit Schlüsseln angegeben, der lexikographisch geordnet ist. In Abb. 3 ist die Ordnung innerhalb der Knoten nach dem in 3.B) beschriebenen Verfahren angegeben. Hierbei ist zu beachten dass durch die Codierung der Baumstruktur im Bitstrom immer zunächst der Vaterknoten und dann die Kindknoten übertragen werden. Desweiteren sind alle Schlüsseleinträge beim Aufbau des Baumes sortiert worden, so dass größer bzw. kleiner Entscheidungen anhand der Schlüsselwerte jedes Knotens getroffen werden kann. Bei der Verwendung von XPATH-Pfaden sind dies beispielsweise ein lexikographische Ordnung entsprechend Beispiel in Abb. 3. Die Schlüssel jedes Knotens im Indexbaum entsprechen ebenfalls dieser Ordnung.

Die ersten Schlüssel in den ersten zwei Kindknoten werden nun relativ zu dem ersten Schlüssel im Vaterknoten codiert. Aufgrund der Sortierung ist der XPATH-Pfad im ersten Schlüssel (ACC s. Abb. 3) des zweiten Kindeknotens häufig sehr ähnlich zum ersten XPATH-Pfad (ACB s. Abb. 3) im Vaterknoten. Jedoch nicht der erste XPATH-Pfad (AA s. Abb. 3) des ersten Kindknotens. Hier ist eher der letzte XPATH-Pfad (AC s. Abb. 3) dem ersten XPATH-Pfad im Vaterknoten sehr ähnlich. Dementsprechend ist die relative Pfadcodierung hier häufig aufwendig.

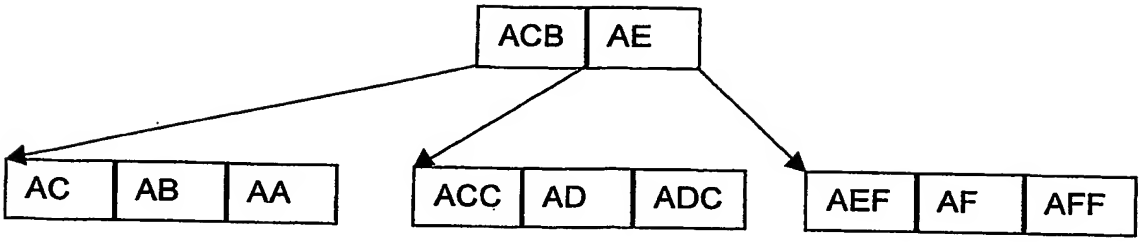


Abb. 6

Um nun auch im ersten Kindknoten eines Vaterknotens des Indexbaums die relative Adressierung effizient einzusetzen muss hier die Codierungsreihenfolge umgekehrt werden. Dies ist in Abb. 6 dargestellt. Nun wird auch im ersten Kindknoten zunächst der Schlüssel codiert, der aufgrund der Ordnung dem bereits übertragenen, ersten Schlüssel des Vaterknotens am ähnlichsten ist.

3.E) Binäre Schlüsselcodierung:

In einer weiteren Ausprägung des in 3.B), 3.C) und 3.D) beschriebenen, erfindungsmäßigen Verfahrens werden als Schlüssel XPATH Pfade verwendet, wobei diese nach dem in [3] spezifizierten Verfahren binär codiert werden.

3.E.1) Ausführung

Im MPEG7 wurde auch eine Binärcodierung von Pfaden standardisiert. Dabei erhält jeder Schritt im Pfad abhängig von Schemainformationen, der Position usw. einen binären Code. Der Einsatz von binär codierten Pfaden kann im Vergleich zur textuellen Repräsentation zu Einsparungen bei der Anzahl der zu übermittelnden Bits führen.

4. Vorteile:

Prinzipiell vermeidet die Einführung eines Indexes das unnötige Decodieren von nicht benötigten Fragmenten. Durch den Einsatz eines B-Baumes kann ein logarithmisches Suchverhalten

erzielt werden. Beide genannten Punkte wirken sich positiv auf die benötigte Suchzeit aus.

Die dargestellte Abbildung eines B-Baumes auf einen linearen Strom trägt dazu bei, dass Indexinformationen einfach zwischen mehreren Beteiligten ausgetauscht werden können. Dabei ist die Übertragung unabhängig von verwendeten Plattformen, Programmiersprachen oder ähnliches. Der Empfänger braucht sich durch die eingefügten Sprunginformationen auch nicht um einen korrekten Aufbau des Baumes kümmern. D. h. ihn interessiert beispielsweise die Ordnung der Knoten im B-Baum nicht.

Des Weiteren haben die eingebauten Sprunginformationen den Vorteil, dass der benötigte Teil des B-Baumes relativ aufgebaut werden kann. Außerdem können für eine Abfrage uninteressante Teilbäume im B-Baum einfach übersprungen werden.

Der Einsatz von relativen Pfaden bzw. die binäre Pfadcodierung führt dazu, dass der zu übertragende Strom möglichst klein gehalten wird.

Literaturverzeichnis:

[1] http://dx1.hrz.uni-dortmund.de:8001/doc1/hrz/sqlref/sqloracle.html

[2] http://www.w3.org/TR/xpath

[3] ISO/IEC 15938-1 Multimedia Content Description Interface - Part 1: Systems, Geneva 2002

[4] TV-Anytime Specification Series S-3 on Metadata, Part-B, Version13.

[5] http://cis.stvincent.edu/carlsond/swdesign/btree/btree.html

[6] http://www.generation5.org/simple_search.shtml

Patentansprüche

1. Verfahren zur Erzeugung eines Bitstroms aus einem Indizierungsbaum, wobei der Indizierungsbaum eine Vielzahl von Knoten in mehreren Hierarchieebenen umfasst, wobei die Knoten Indexdaten umfassend Schlüssel enthalten und die Indexdaten bezüglich der Schlüssel sortiert sind, bei dem:

   - die Indexdaten der Knoten in den Bitstrom eingefügt werden;

   - wobei beim Einfügen der Indexdaten eines Knotens in einer Hierarchieebene im Bitstrom die Information gespeichert wird, an welcher Stelle im Bitstrom sich die Indexdaten von einem oder mehreren Knoten der darunter liegenden Hierachieebene befinden.

2. Verfahren nach Anspruch 1, bei dem die Indexdaten nach der Depth-First-Ordnung in den Bitstrom eingefügt werden.

3. Verfahren nach einem der vorhergehenden Ansprüche, bei dem die Schlüssel XPATH-Pfade sind.

4. Verfahren nach einem der vorhergehenden Ansprüche, bei dem die Schlüssel der Indexdaten eines Knotens in einer Hierarchieebene relative XPATH-Pfade in Bezug auf den zuvor im Bitstrom eingefügten Schlüssel des gleichen Knotens oder eines Knotens der höheren Hierarchieebene sind.

5. Verfahren nach Anspruch 3 oder 4, bei dem die XPATH-Pfade mit einem Codierverfahren binär codiert werden, insbesondere mit einem MPEG-Codierverfahren.

6. Verfahren nach Anspruch 5, bei dem das Codierverfahren ein MPEG7-codierverfahren ist.

7. Verfahren nach einem der vorhergehenden Ansprüche, bei dem sich die Indexdaten auf Beschreibungselemente eines XML-Dokuments beziehen.

8. Verfahren zum Codieren einer Datenstruktur, bei dem die Datenelemente der Datenstruktur in einem Indizierungsbaum indiziert sind, wobei ein Bitstrom gemäß einem der vorhergehenden Ansprüche erzeugt wird und der Bitstrom Teil des codierten Datenstroms ist.

9. Verfahren zum Decodieren einer Datenstruktur, wobei das Verfahren derart ausgestaltet ist, dass die gemäß Anspruch 8 codierte Datenstruktur decodiert wird.

10. Verfahren zum Codieren und Decodieren einer Datenstruktur, umfassend das Verfahren nach Anspruch 8 und das Verfahren nach Anspruch 9.

11. Codiervorrichtung, mit der ein Verfahren nach Anspruch 8 durchführbar ist.

12. Decodiervorrichtung mit der ein Verfahren nach Anspruch 9 durchführbar ist.

13. Vorrichtung zum Codieren und Decodieren einer Datenstruktur, mit der ein Verfahren nach Anspruch 10 durchführbar ist.